

# LARSOFT OPTICAL PHYSICS SIMULATIONS

---

Ben Jones  
MIT

# This Talk

- Summarize the current implementation of optical simulation tools in LArSoft
- Also the physics list infrastructure which was put in place to accommodate them (per Eric's request)
- Describe ongoing work on LArSoft photon propagation tools and scintillation simulation
- Discussion of natural hook-in points for alternative optical simulation strategies (NEST, etc)

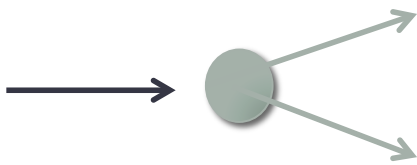
# Contents of the Talk

- 1 – Physics List Handling in LArSoft**
- 2 – The Full Optical Simulation Physics Objects**
- 3 – Light Sources, Analyzers and Geometry**
- 4 – Results from MicroBooNE Studies**
- 5 – Fast Simulations**
- 6 – What MicroBooNE still needs to do,  
and what LBNE still needs to do**
- 7 – Hooks for Alternative Treatments of Optical Physics**

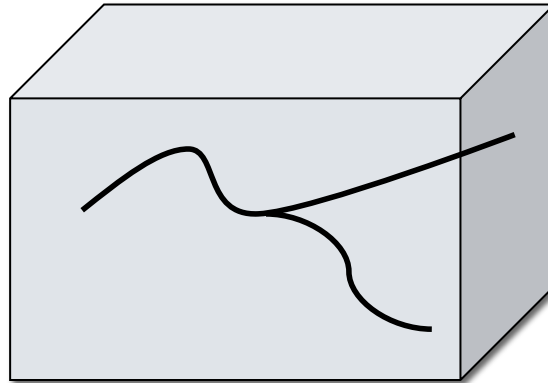
# 1 – Physics Lists in LArSoft

# Physics Lists

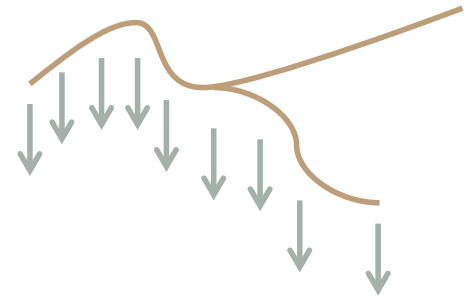
- A physics list is an object in LArG4 which tells Geant4 which physics processes to enable when particles are stepping through the detector volume.



1. Neutrino nucleus interaction (event generator)



2. Stepping particles through detector (LArG4)



3. Charge drift, electronics simulations, etc

# What Does a Physics List Do?

- A physics list defines all particles and interactions which Geant4 can track.
- Every process attached to a particle has a characteristic step length, this determines how far each step in the simulation is
- Hence no physics processes = no particle stepping
- Particles and interactions are organized into smaller sub-units called physics constructors.
- The job of the physics list is to register the appropriate set of physics constructors and hence select the required particles and interactions for the simulation

# QGSP\_BERT

```
#include "G4EmStandardPhysics.hh"
#include "G4EmExtraPhysics.hh"
#include "G4IonPhysics.hh"
#include "G4QStoppingPhysics.hh"
#include "G4HadronElasticPhysics.hh"
#include "G4NeutronTrackingCut.hh"

#include "G4DataQuestionnaire.hh"
#include "HadronPhysicsQGSP_BERT.hh"

template<class T> TQGSP_BERT<T>::TQGSP_BERT(G4int ver): T()
{
    G4DataQuestionnaire it(photon);
    G4cout << "<<< Geant4 Physics List simulation engine: QGSP_BERT 3.3"<<G4endl;
    G4cout <<G4endl;

    this->defaultCutValue = 0.7*mm;
    this->SetVerboseLevel(ver);

    // EM Physics
    this->RegisterPhysics( new G4EmStandardPhysics("standard EM",ver));

    // Synchrotron Radiation & GN Physics
    this->RegisterPhysics( new G4EmExtraPhysics("extra EM"));

    // Decays
    this->RegisterPhysics( new G4DecayPhysics("decay",ver) );

    // Hadron Elastic scattering
    this-> RegisterPhysics( new G4HadronElasticPhysics("elastic",ver,false));

    // Hadron Physics
    G4bool quasiElastic;
    this->RegisterPhysics( new HadronPhysicsQGSP_BERT("hadron",quasiElastic=true));

    // Stopping Physics
    this->RegisterPhysics( new G4QStoppingPhysics("stopping"));

    // Ion Physics
    this->RegisterPhysics( new G4IonPhysics("ion"));

    // Neutron tracking cut
    this->RegisterPhysics( new G4NeutronTrackingCut("Neutron tracking cut", ver));
}
```

# Physics Constructors

- The first thing a physics constructor does is declare all the particles it applies to
- Then for each type of particle, a singleton process manager is passed instructions on which physics processes apply to that particle
- For example, some lines from G4EmPhysicsStandard:

```
00118 // gamma
00119 G4Gamma::Gamma();
00120
00121 // leptons
00122 G4Electron::Electron();
00123 G4Positron::Positron();
00124 G4MuonPlus::MuonPlus();
00125 G4MuonMinus::MuonMinus();
.....

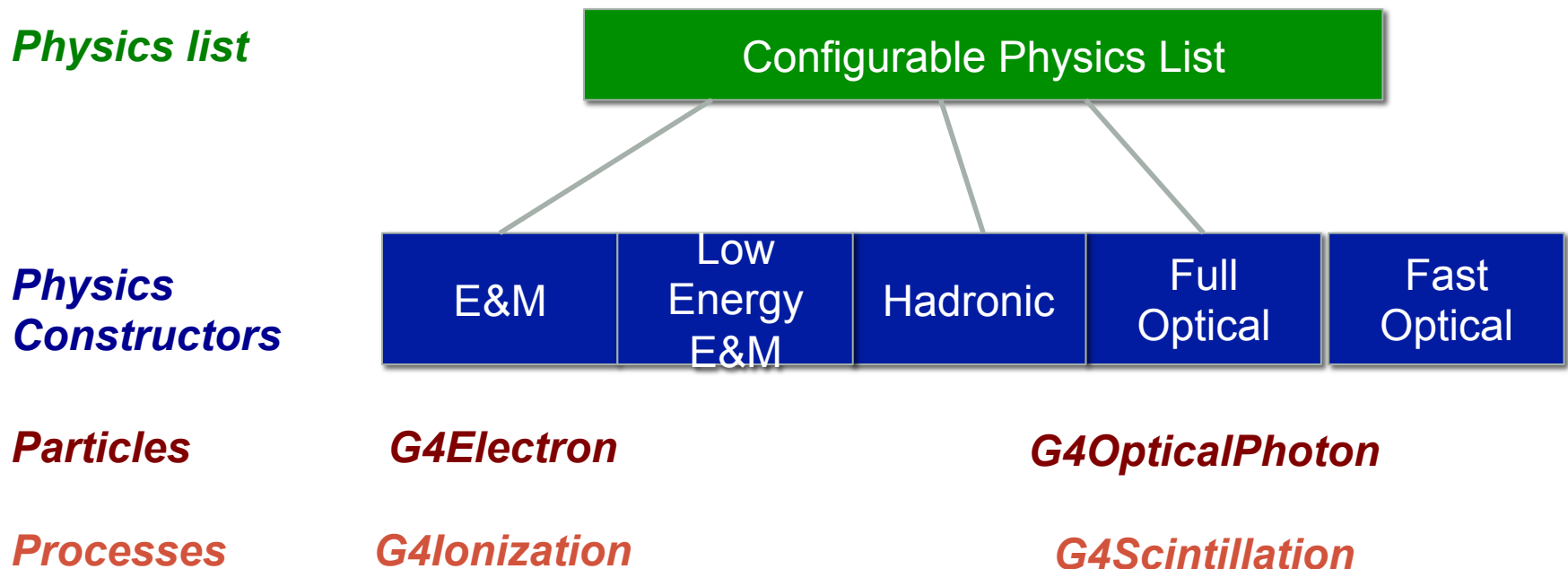
00160 if (particleName == "gamma") {
00161
00162 pmanager->AddDiscreteProcess(new
G4PhotoElectricEffect);
00163 pmanager->AddDiscreteProcess(new
G4ComptonScattering);
00164 pmanager->AddDiscreteProcess(new
G4GammaConversion);
00165
00166 } else if (particleName == "e-") {
00167
```



# Configurability of LArSoft physics list

- For most simulation applications, a basic physics list is chosen and then tweaked to meet the needs of the experiment at hand.
- In LArSoft we need some more flexibility – for some simulation jobs we want to simulate optical photons, but for others this is unnecessary
- Running optical simulations carries a high computational pricetag – a few hours per event, so when and how to run it varies job to job

## *Physics list*





# Setting the Physics List

- The LArSoft physics list is built to allow users to turn on or off any Geant4 physics constructor at run time.
- It works like this:
- Exerpt from simulationservices.fcl:

```
UseCustomPhysics:      false  #Whether to use a custom list of physics processes or the default
EnabledPhysics:        [ "Em", "SynchrotronAndGN", "Ion", "Hadron",
                          "Decay", "HadronElastic", "Stopping", "NeutronTrackingCut" ]
```

Names of physics constructors ^^

# Registering a module

- No code added to the physics list class to add new module, rather:
- Custom physics class:

*Eg in OpticalPhysics.cxx*

```
CustomPhysicsFactory<OpticalPhysics> optical_factory("Optical");
```

- Existing Geant4 physics class are wrapped in CustomPhysicsFactories in the files

*CustomPhysicsBuiltIns.hh*

*CustomPhysicsBuiltIns.cxx*

# Adding a custom physics constructor



## Using Custom Physics Modules:

LArG4 now contains a configurable physics list which allows the user to enable or disable physical processes used in the GEANT4 simulation. To control which G4PhysicsConstructors are loaded, set the following two parameters for the LArG4Parameters service, eg:

```
UseCustomPhysics      = larg4.bool(False),  
EnabledPhysics        = larg4.vstring( 'Em' 'Optical' 'SynchotronAndGN' 'Ion' 'Hadron' 'Decay' 'HadronElastic' '
```

The default list of physics processes, as included in the QGSP\_BERT physics list (the previous default before the list was configurable), are:

```
"Em" "SynchotronAndGN" "Decay" "Hadron" "HadronElastic" "Stopping" "Ion" "NeutronTrackingCut"
```

To create a new physics constructor, create a class which inherits from G4VPhysicsConstructor, providing the necessary ConstructParticle and ConstructProcess methods (see a pre-existing GEANT4 physics constructor for an example). Then register the object in the physics list at compile time by including the following line at the top of the .cxx file for the object:

```
CustomPhysicsFactory<Object> arbitrary_factory_name("ObjectName");
```

Where Object is the name of the object inheriting from G4VPhysicsConstructor, arbitrary\_factory\_name is an irrelevant label for the object which registers the new physics constructor and ObjectName is the string which will be used in the job control file to enable the physics processes.

The default constructors loaded in the QGSP\_BERT physics list are all registered in the CustomPhysicsBuiltIns.hh and CustomPhysicsBuiltIns.cxx files. Under no circumstances should new modules be registered this way - this file just provides wrappers for the default GEANT4 objects. To register a new physics constructor, use the method described above.

# 2 – Full Optical Sim

## Physics Objects

## root / trunk / LArG4

### Name

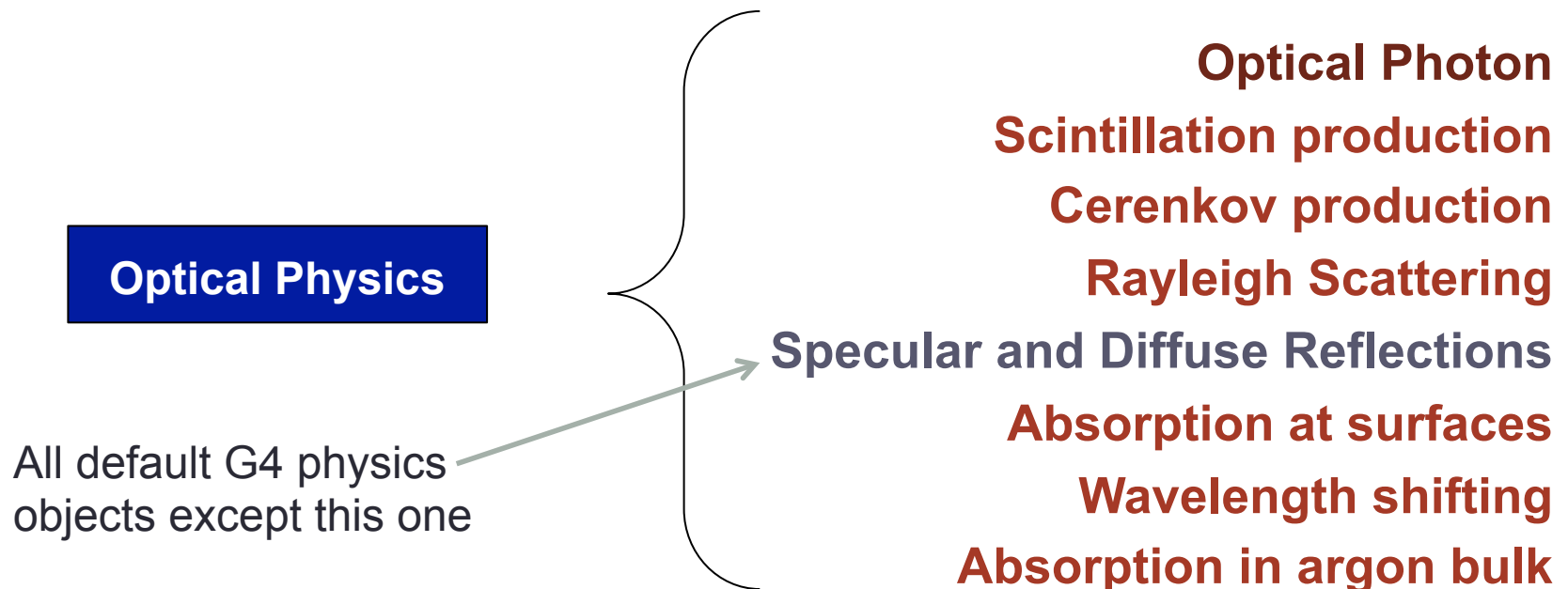
ConfigurablePhysicsList.hh ←  
ConfigurablePhysicsList.icc ←  
CustomPhysicsBuiltIns.cxx ←  
CustomPhysicsBuiltIns.hh ←  
CustomPhysicsFactory.hh ←  
CustomPhysicsTable.cxx ←  
CustomPhysicsTable.hh ←  
G4BadIdeaAction.cxx  
G4BadIdeaAction.h  
GNUmakefile  
LArG4.cxx ← ←  
LArG4.h ← ←  
LArG4.mac  
LArG4Ana.cxx  
LArG4Ana.h  
LArG4Ana\_module.cc  
LArG4\_module.cc  
LArStackingAction.cxx  
LArStackingAction.h  
LArVoxelReadout.cxx  
LArVoxelReadout.h  
LArVoxelReadoutGeometry.cxx  
LArVoxelReadoutGeometry.h  
MaterialPropertyLoader.cxx ←  
MaterialPropertyLoader.h ←

MuNuclearSplittingProcess.cxx  
MuNuclearSplittingProcess.h  
MuNuclearSplittingProcessXSecBias.cxx  
MuNuclearSplittingProcessXSecBias.h  
OpBoundaryProcessSimple.cxx ←  
OpBoundaryProcessSimple.hh ←  
OpticalPhysics.cxx ←  
OpticalPhysics.hh ←  
PMTLookup.cxx ←  
PMTLookup.h ←  
PMTReadoutGeometry.cxx ←  
PMTReadoutGeometry.h ←  
PMTSensitiveDetector.cxx ←  
PMTSensitiveDetector.h ←  
ParticleListAction.cxx  
ParticleListAction.h  
PhysicsList.cxx ←  
PhysicsList.h ←  
VisualizationAction.cxx  
VisualizationAction.h  
atree.mac  
largeantmodules.fcl

Physics List Code  
Optical Code

# Optical Processes in LArG4

- The physics list in LarSoft is configurable, with various physics constructors which can be switched on and off at job configuration time
- Optical physics processes are loaded via the "OpticalPhysics" GEANT4 physics constructor, which was customized to fit our needs in LarSoft.





# Optical Properties of Materials

- Optical properties of materials are loaded during the detector construction step using the MaterialPropertyLoader class.
- The requirement of loading wavelength dependent parameters required us to step outside the default gdml parser and implement this new class.
- This

## Per Material Type

Scintillation
<i>Fast component spectrum</i> <i>Slow component spectrum</i> <i>Scintillation yield</i> <i>Fast time const</i> <i>Slow time const</i> <i>Proportion fast / slow</i> <i>Quenching per particle</i>

Cerenkov
- none -

Absorption
<i>Absorption Length</i>
Rayleigh Scattering
<i>Scattering Length</i>

WLS
<i>Absorption spectrum</i> <i>Emission spectrum</i> <i>Time Constant</i> <i>Yield out / in</i>

## Per Boundary Type

Reflections
<i>Total Reflectivity</i> <i>Fraction specular / diffuse</i>

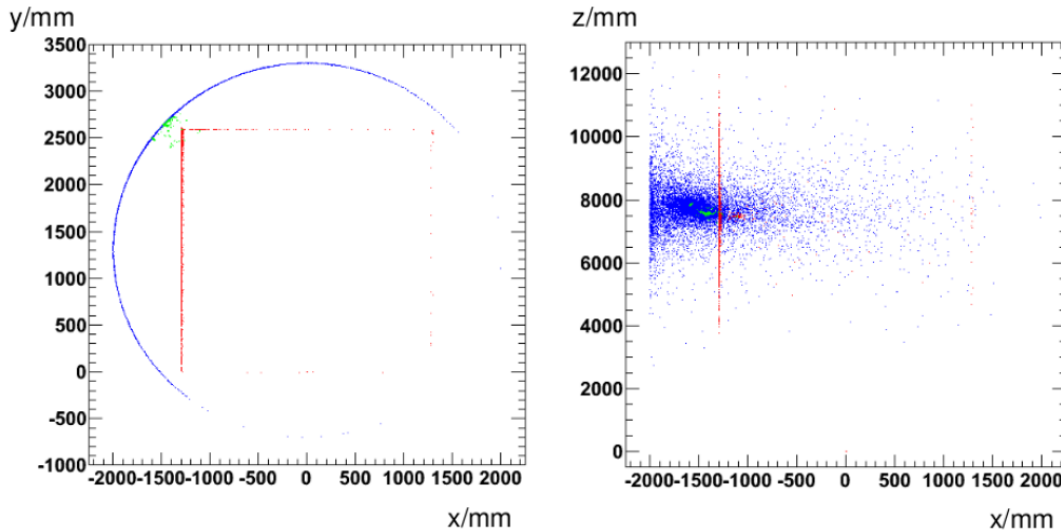
<i>Wavelength dependent</i> <i>Non wavelength dependent</i>
--

# Optical Material Properties

- GDML is not adaptable enough to supplying relevant material properties in the detector geometry, so a LArSoft class handles this
- Optical properties of liquid argon are supplied by the LArProperties LArSoft service.
- At config time, a table of all specified properties is built in LArSoft.
- Then immediately after detector construction, run through the volumes attaching relevant properties.
- This must happen before construction of the physics list, or the optical processes will not configure properly.

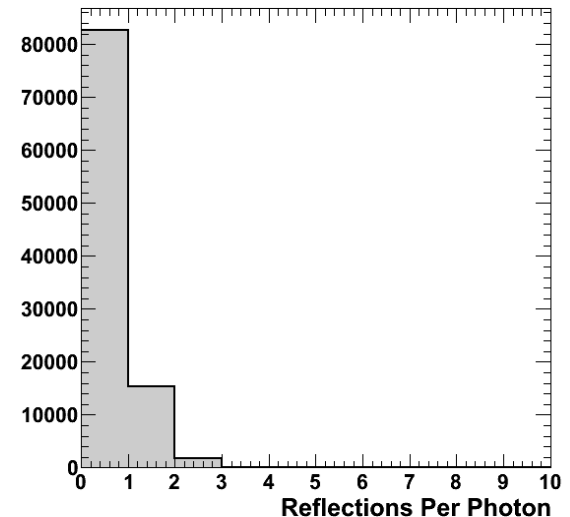
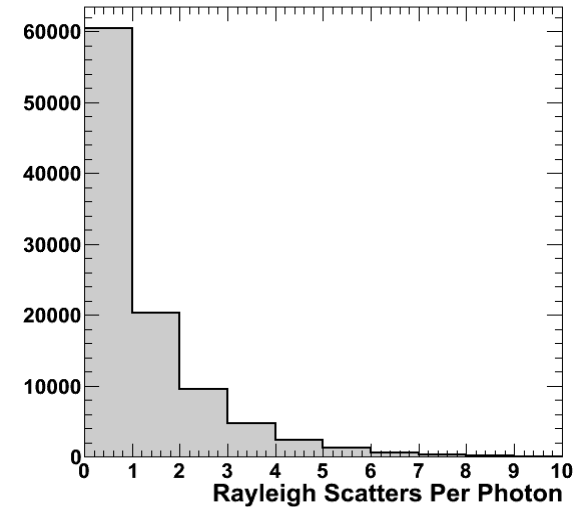
```
22
23 FastScintEnergies: [ 9.5, 9.7, 9.9]
24 FastScintSpectrum: [ 0.5, 1.0, 0.5]
25 SlowScintEnergies: [ 9.5, 9.7, 9.9]
26 SlowScintSpectrum: [ 0.5, 1.0, 0.5]
27
28 RIndexEnergies:    [ 9.5, 9.7, 9.9]
29 RIndexSpectrum:    [ 1.38, 1.38, 1.38]
30 AbsLengthEnergies: [ 9.5, 9.7, 9.9]
31 AbsLengthSpectrum: [ 2000., 2000., 2000.]
32 RayleighEnergies:  [ 9.5, 9.7, 9.9]
33 RayleighSpectrum:  [ 90., 90., 90.]
34
35 ScintYield:         24000.
36 ScintResolutionScale: 0.005
37 ScintFastTimeConst: 6
38 ScintSlowTimeConst: 1590.
39 ScintYieldRatio:    0.3
40 ScintBirksConstant: 0.00322
41
42 ReflectiveSurfaceNames:          [ "STEEL_STAINLESS_Fe7Cr2Ni" ]
43
44 ReflectiveSurfaceEnergies:        [ 9.5, 9.7, 9.9 ]
45 ReflectiveSurfaceReflectances:    [ [ 0.25, 0.25, 0.25 ] ]
46 ReflectiveSurfaceDiffuseFractions: [ [ 0.5, 0.5, 0.5 ] ]
47
48 }
```

# A Sample Neutrino Event in LArG4



**Green** - **Photon production**  
**Blue** - **Photon absorption at surface of known reflectivity**  
**Red** - **Photon absorption at surface with no reflectivity data**

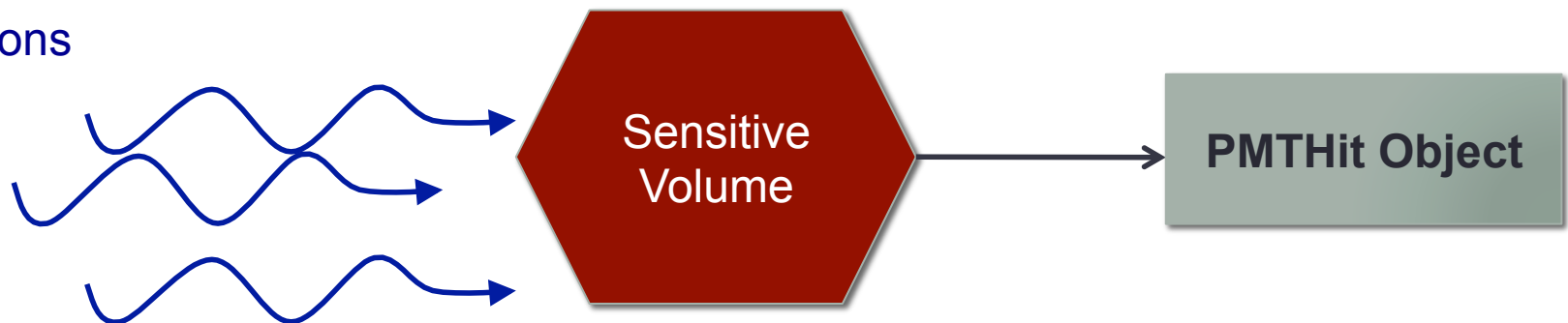
- 95161 photons were generated of which 68996 were eventually absorbed at a steel surface and 20932 were absorbed into a "black area"
- Each photon underwent a mean of 0.76 Rayleigh scatters and 0.19 reflections



# Photon Stepping and Detection

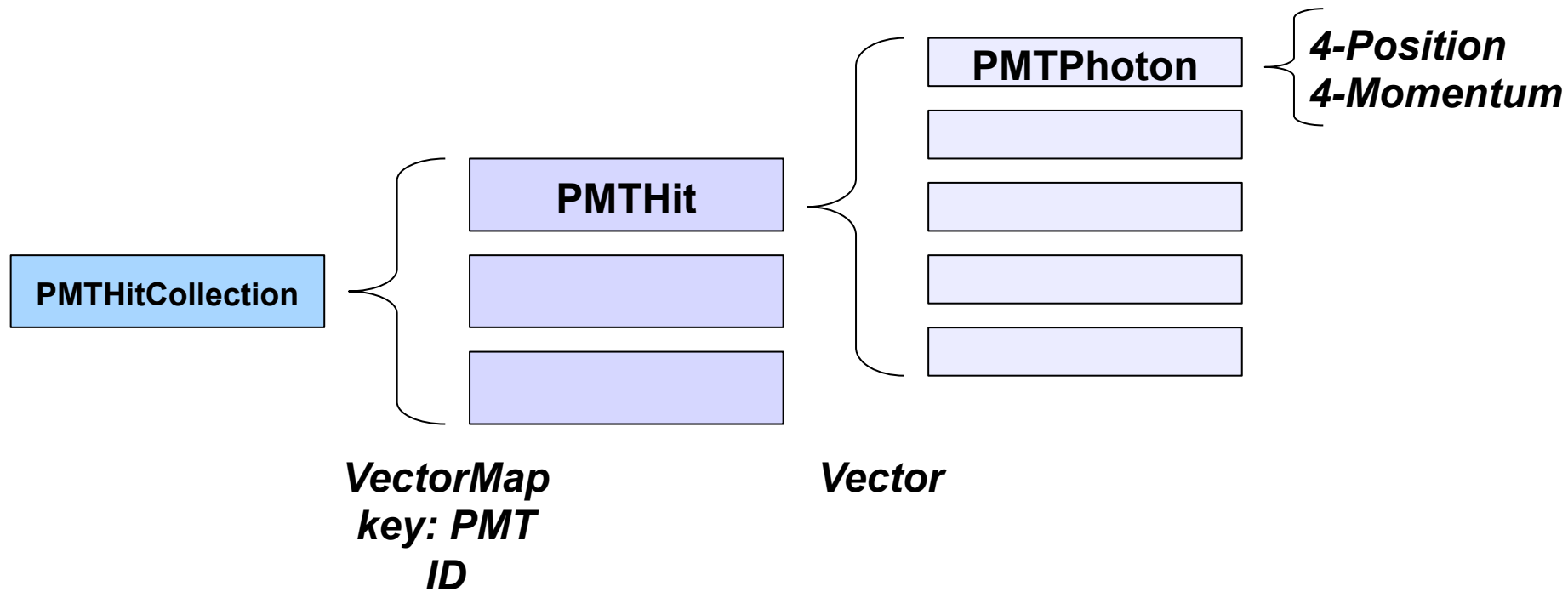
- Optical Photon is stepped like any other Geant4 particle
- Weight the number at production by the quantum efficiency of the detection device – no need to step all of them if only 3% QE
- A PMTSensitiveDetector object is responsible for figuring out when PMTs step into volumes designated as sensitive
- These are set up during the geometry build, specified by volume name in the LArG4 config
- This is completely detector agnostic, and reports data on all photons stepping into its volume. Information is recorded on location the photon stepped in, its angle and wavelength, for use in digitization stage

Photons



# PMTHit Data Structures

- The optical information to be passed along the simulation chain from LArG4 is contained within a PMT hit collection
- The PMTHitCollection is a set of PMTHits, one for each PMT that saw one or more photon
- Each hit is a list of 4-positions and 4 momenta of photons which stepped across the lens of the PMT

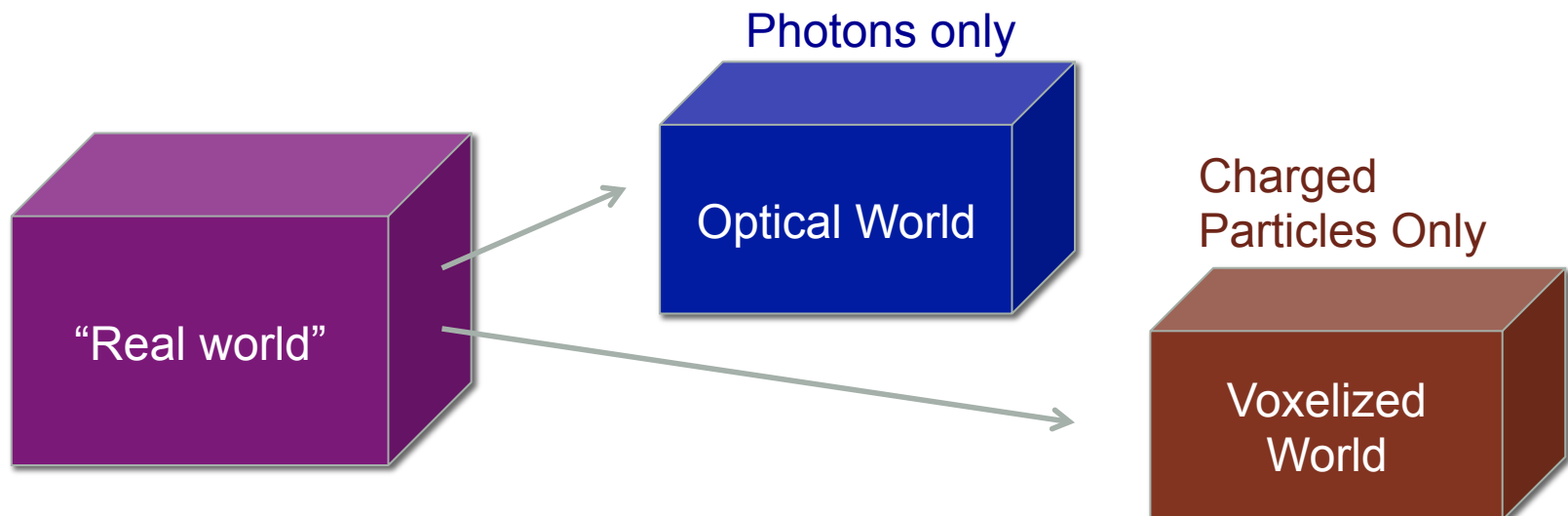


# Technical Details – Readout Geometry and Sensitive Detectors

- In fact, to produce a coherent PMTHitCollection in the event, there is only one PMTSensitiveDetector.
- Also, sensitive objects may be a child volume of a replicated parent volume. This means that only one copy exists in memory, and sensitive volumes which receive a photon can't explicitly tell which PMT they are a part of.
- This is solved via the PMTReadoutGeometry object, which derives from G4VUserParallelWorld.
- This object has an individual geometrical object per PMT sensitive object placement, and hooks up to a G4SensitiveDetector.
- This follows a standard G4 design pattern, and there are comments in the code.
- PMT ID to volume matching are handled by an object called PMTLookup, so each sensitive element can figure out its PMTID when it needs to.

# Geometry Parallelization

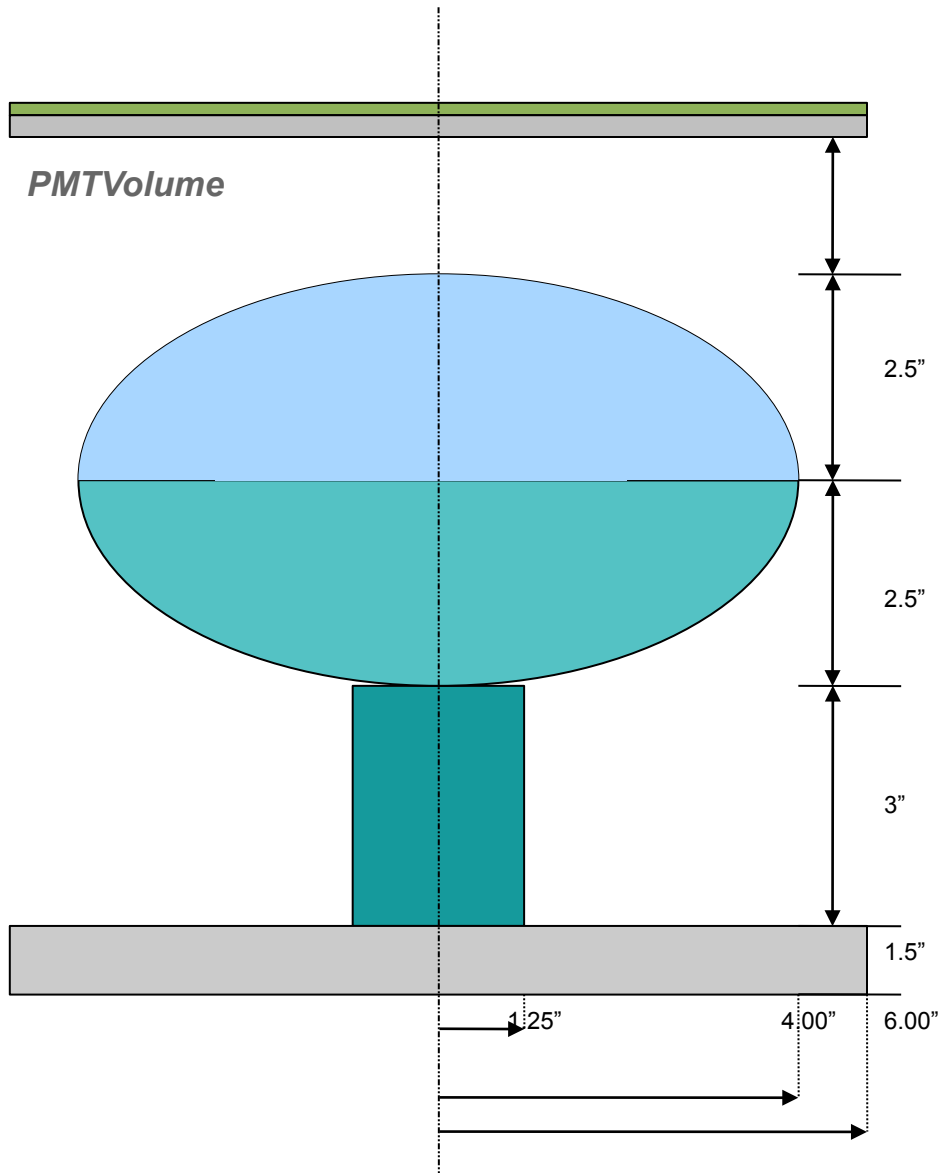
- To prevent photons from being aware of voxel boundaries in the LAr bulk, they exist in a parallel Geant4 geometry
- This geometry is also where sensitive photon detectors live – no particle other than photons can see them
- This is defined analogously to the LArVoxel parallel geometry



# 3 – Light Sources, Analyzers and Geometry



# PMTs in LArSoft



Two ways to run the PMT simulation:

***Either:***

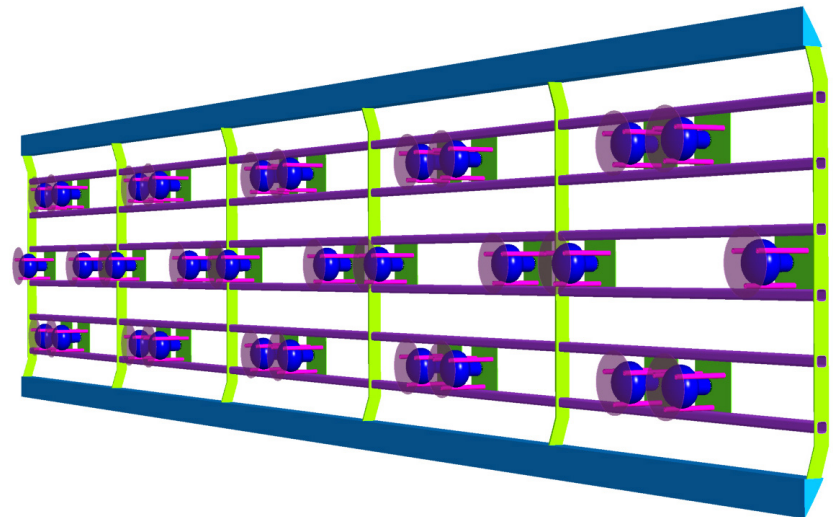
**1 :** Attach the WLS process to the TPB plate and make the PMT lens volume sensitive

**2 :** Make the PMT plate volume sensitive and factor everything else into an overall efficiency of the PMT assembly

At the moment, we favour approach number 2

# PMT Placement and Geometry

- The geometry files used for LArSoft experiments are written in the **GDML** language and built using a set of geometry generation scripts
- **PMT geometry** definition and placement scripts have been added
- **microboone.gdml** has been rebuilt with coordinates from one possible 30 PMT design
- PMTs are placed by supplying the **x,y,z coordinates of the centre of the PMT lens ellipsoid** and the **direction of the lens normal**
- During geometry parsing, PMT components are used to build a **parallel world volume** and appropriate **sensitive volumes** with **PMT ID's** are assigned
- The optical simulation has already been used to optimize the design of the PMT rack, choosing a straight over a curved design.



root / trunk / Geometry / gdml / microboone / pmt-coordinates.csv

History | View | Annotate | Download (998 Bytes)

1	142.20,18.42,40.53,7.50,143.30
2	142.20,-18.42,40.53,-7.50,143.30
3	136.50,43.18,106.36,17.50,143.30
4	136.50,-43.18,106.36,-17.50,143.30
5	142.20,18.42,172.19,7.50,143.30
6	142.20,-18.42,172.19,-7.50,143.30
7	142.20,18.42,245.64,7.50,143.30
8	142.20,-18.42,245.64,-7.50,143.30
9	136.50,43.18,311.47,17.50,143.30
10	136.50,-43.18,311.47,-17.50,143.30
11	142.20,18.42,377.30,7.50,143.30

etc

root / trunk / Geometry / gdml / microboone / gen\_pmt.pl

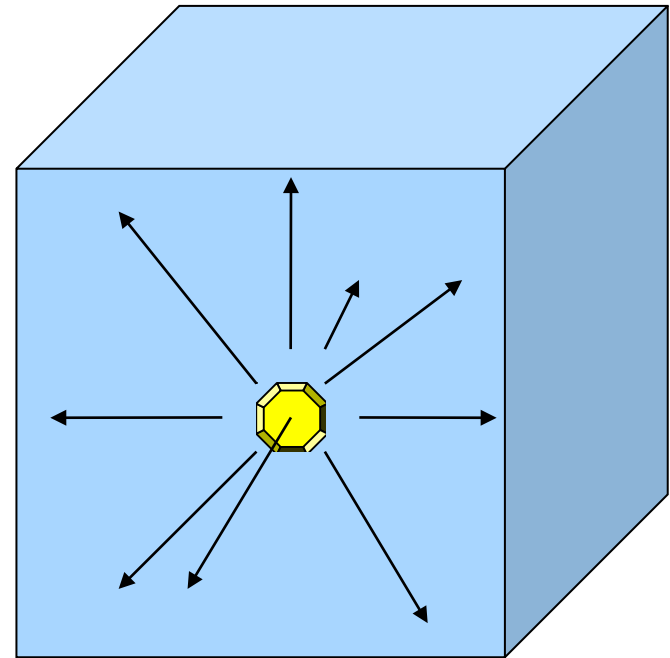
History | View | Annotate | Download (3.1 KB)

```
1 #!/usr/bin/perl
2
3 sub gen_pmt {
4
5     $PMT = "micro-pmtdef" . $suffix . ".gdml";
6     push (@gdmlFiles, $PMT); # Add file to list of GDML fragments
7     $PMT = ">" . $PMT;
8     open(PMT) or die("Could not open file $PMT for writing");
9
10    print PMT <<EOF;
11 <solids>
12 <tube name="PMTVolume"
13   rmax="(6*2.54)"
14   z="(11.0*2.54)"
15   deltaphi="2*(3.1415926535897)"
16   aunit="rad"
17   lunit="cm"/>
18 <tube name="PMT_TPBCoating"
```

etc

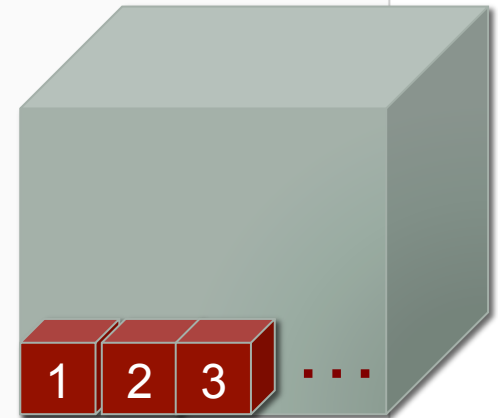
# The Light Source Event Generator

- Event generator which simulates an extended, isotropic light source at some position in the detector
- Light source is a cuboid of specified dimension and location, which can change event by event
- Two ways to systematically steer the light source:
  - **File Mode**  
Event by event specification of source position, size and intensity
  - **Scan Mode**  
Systematically scan cuboidal voxels of a specified size event by event
- Optionally, a data structure can be stored in the event with details of the light source configuration



# Scan Mode

```
24 // SCAN MODE:
25 // Divide volume into cuboidal regions and produce an isotropic light source in each,
26 // using one region per event. User can specify either to use the full detector volume
27 // or some custom specified volume.
28 //
29 // This mode is used when building a fast photon sim library, and performing volume
30 // scan sensitivity studies.
31 //
32 // int32   SourceMode = 1      - sets light source to scan mode
33 // int32   N               - number of photons to shoot from each point
34 // double  P               - peak photon momentum (or energy) in eV
35 // double  SigmaP          - momentum distribution width
36 // double  XSteps          - Number of regions to divide volume into in each direction
37 // double  YSteps
38 // double  ZSteps
39 // double  T0              - Peak time of photon production
40 // double  SigmaT          - time distribution width
41 // int32   PosDist         - how to distribute production points sampled in momentum, position
42 // int32   PDist           and time ranges specified. For all of these :
43 // int32   TDist           0 = uniform and 1 = gaussian
44 // bool    FillTree        - whether to write a tree of photon production points to fileservice
45 // bool    BuildLibrary    - whether to write the light source specification to the event for library building / analysis
46 // bool    UseCustomRegion - supply our own volume specification or use the full detector volume?
47 // vdouble[3] RegionMin    - bounding corners of the custom volume specification
48 // vdouble[3] RegionMax    (only used if UseCustomRegion=true)
49 //
50
51
52
```

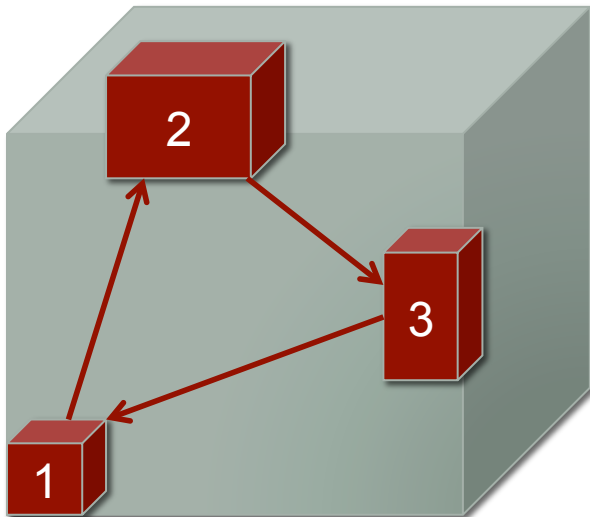


# File Mode

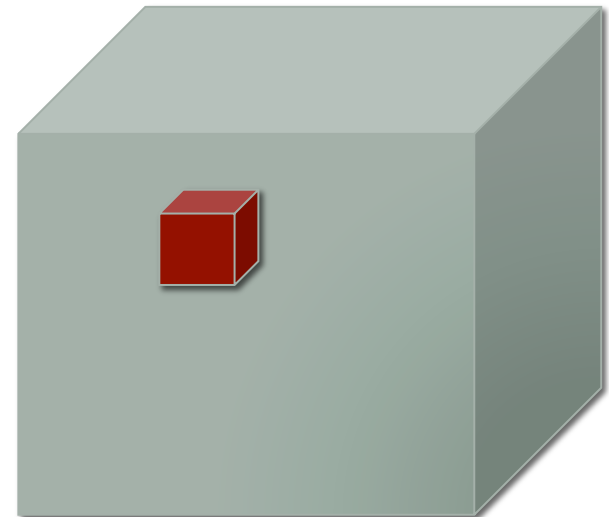
**root / trunk / EventGenerator / LightSource.txt**

[History](#) | [View](#) | [Annotate](#) | [Download \(252 Bytes\)](#)

1	x	y	z	t	dx	dy	dz	dt	p	dp	n
2	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	9.7	0.1	1000
3	1.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	9.7	0.1	1000
4	2.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	9.7	0.1	1000
5	3.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	9.7	0.1	1000
6	4.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	9.7	0.1	1000



Or...

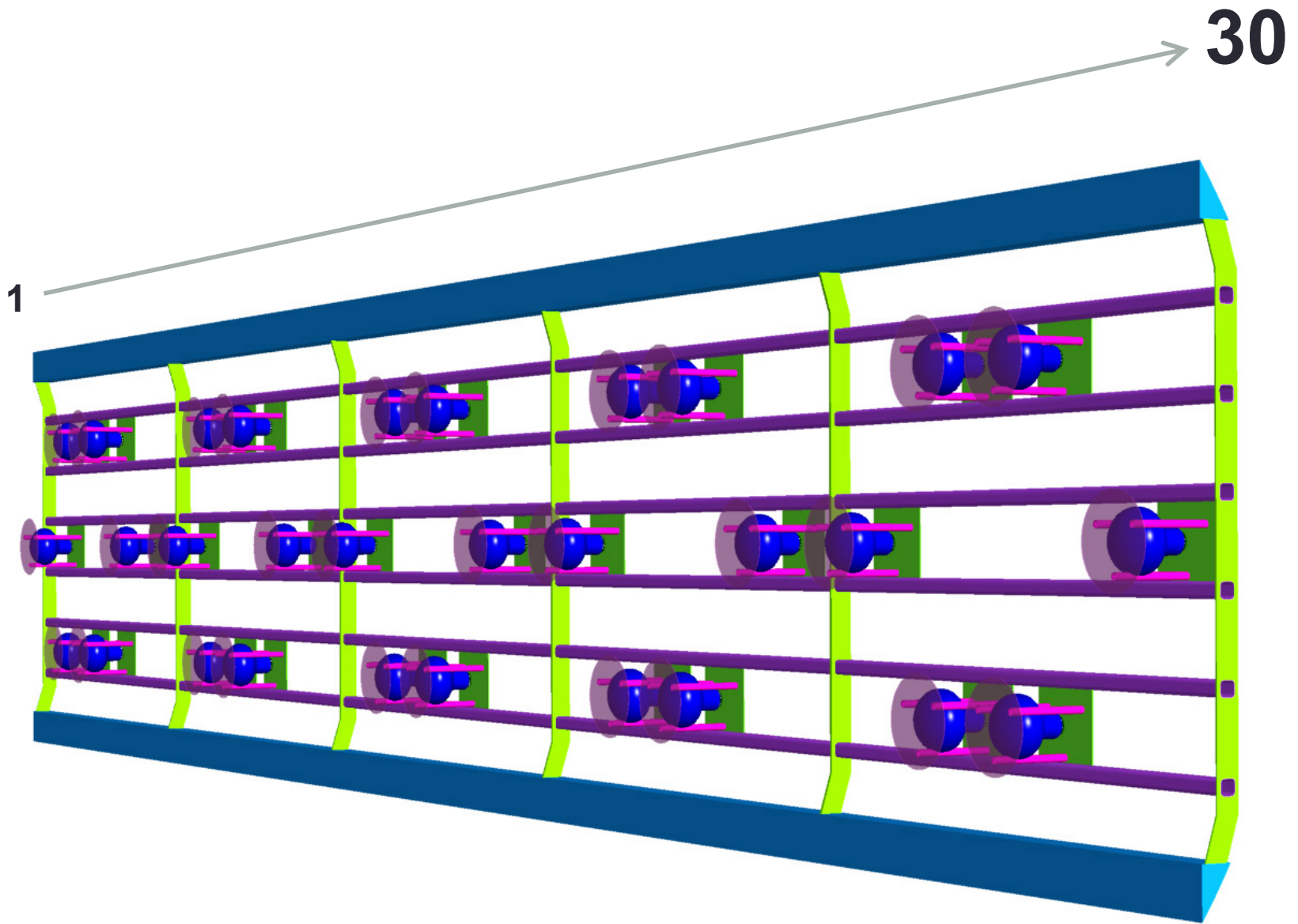


# The PMTResponseAnalyzer

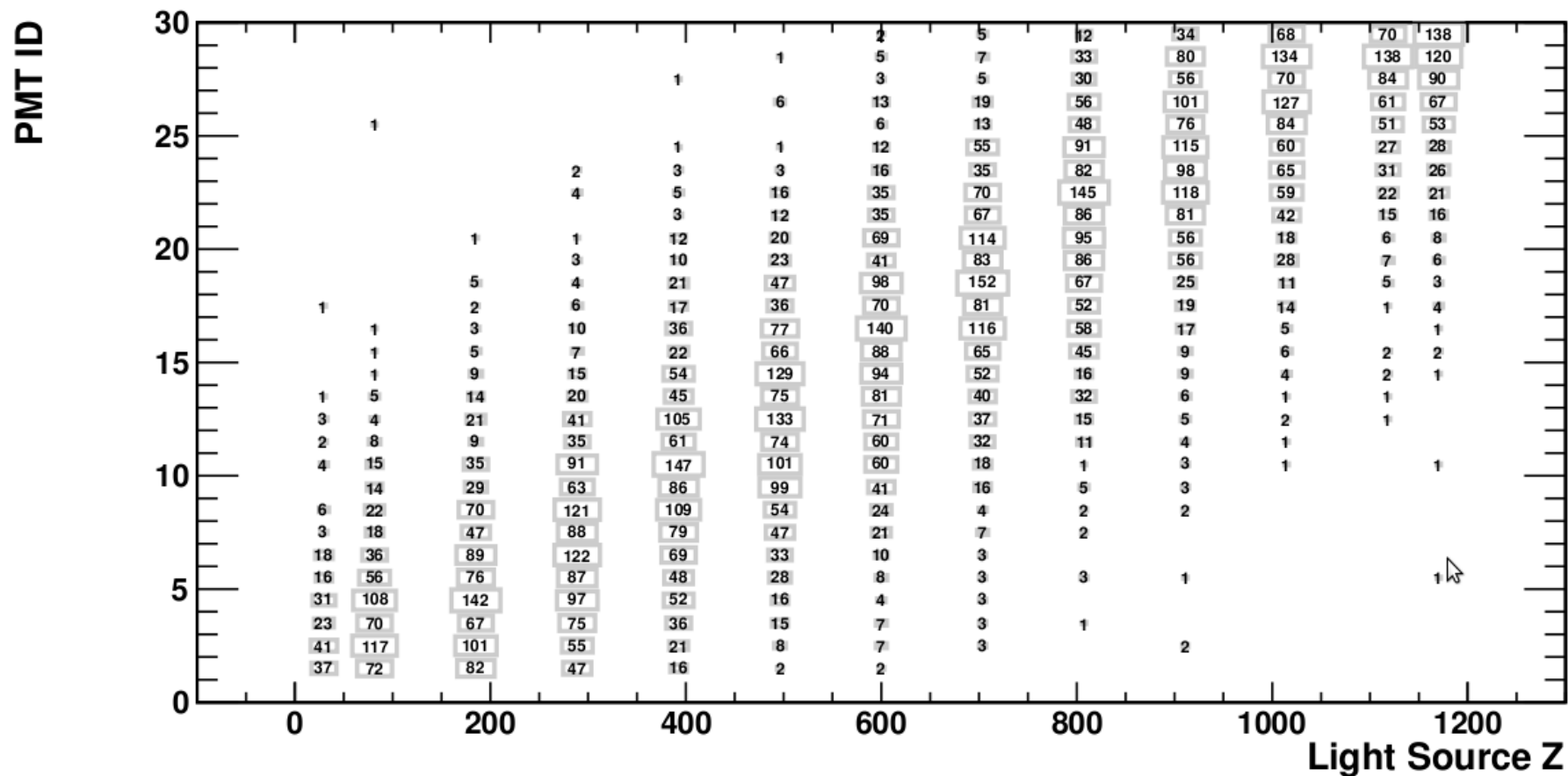
- **A pretty simple analyzer, but very useful:**
- For each event, grab the PMTHitCollection
- Produce TTrees at 4 possible detail levels (user chooses)
  - Full kinematics of every detection candidate photon
  - Full kinematics of randomly selected photons (QE)
  - Number of photons per PMT per event
  - Total number of photons per event
- Optional:
  - High and low wavelength cutoff
  - Poisson sampling according to quantum efficiency

## 4 – Trigger / Sensitivity Studies in MicroBooNE



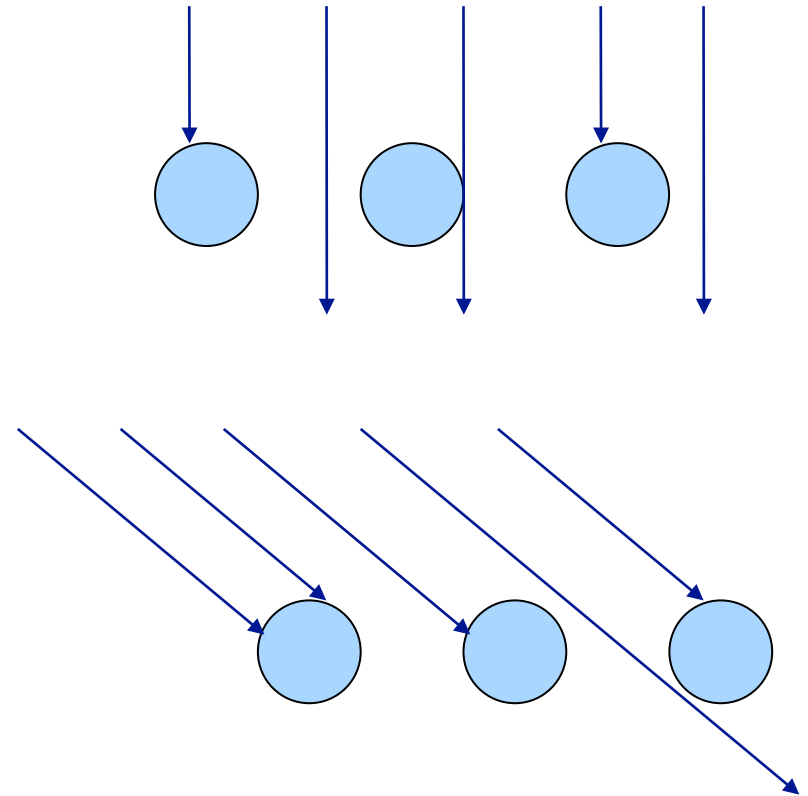
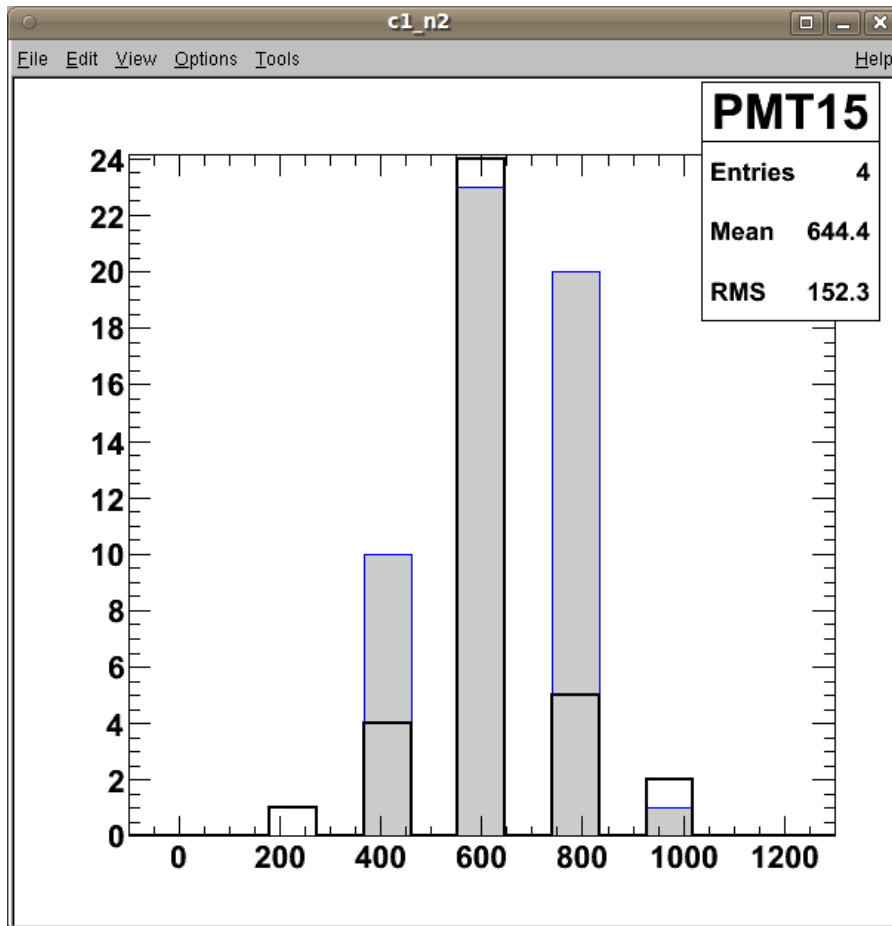


# On Axis Point Source Test

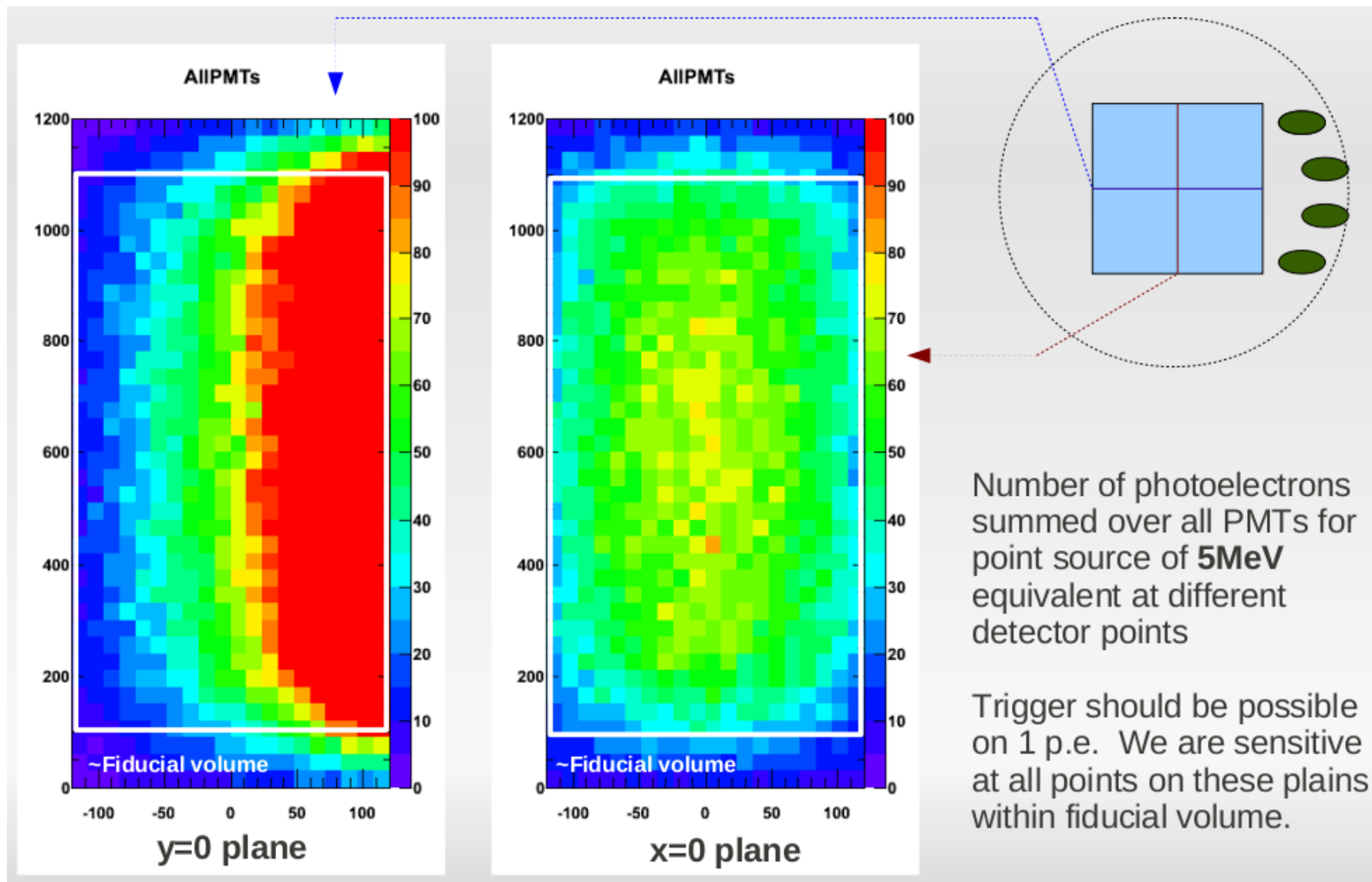


# Effect of Wire Planes

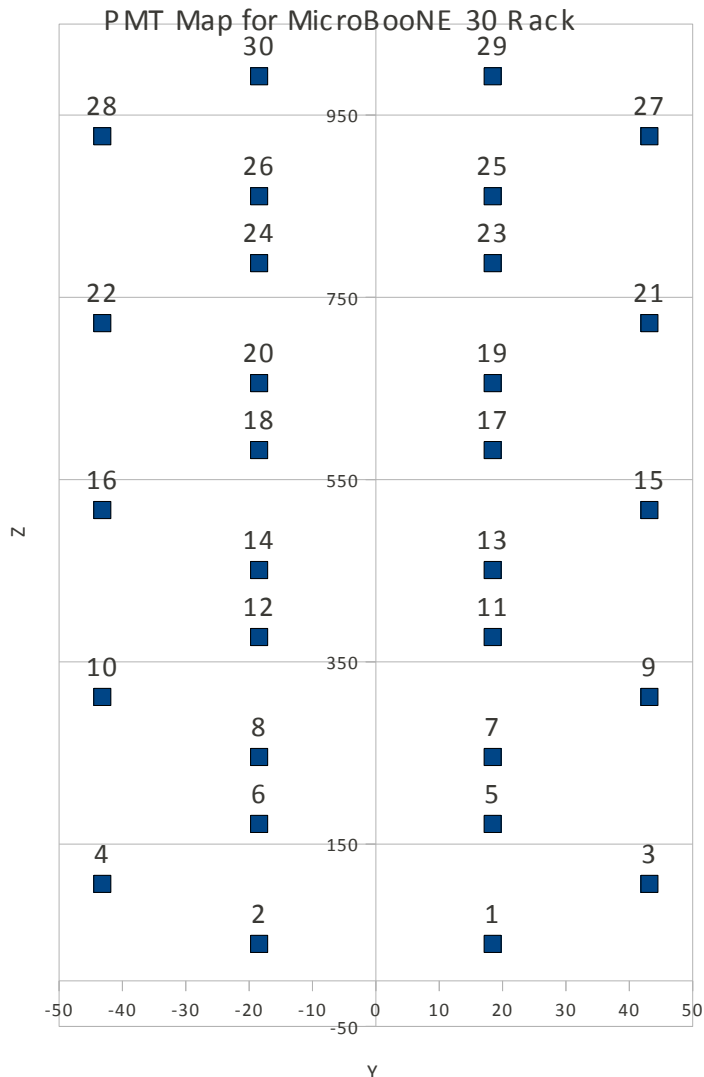
- Considering only one central PMT – note that the large angle light is more strongly blocked. Explains the flattening on the previous slide.



# Sensitivity Maps



# PMT Coverage and Redundancy Tests

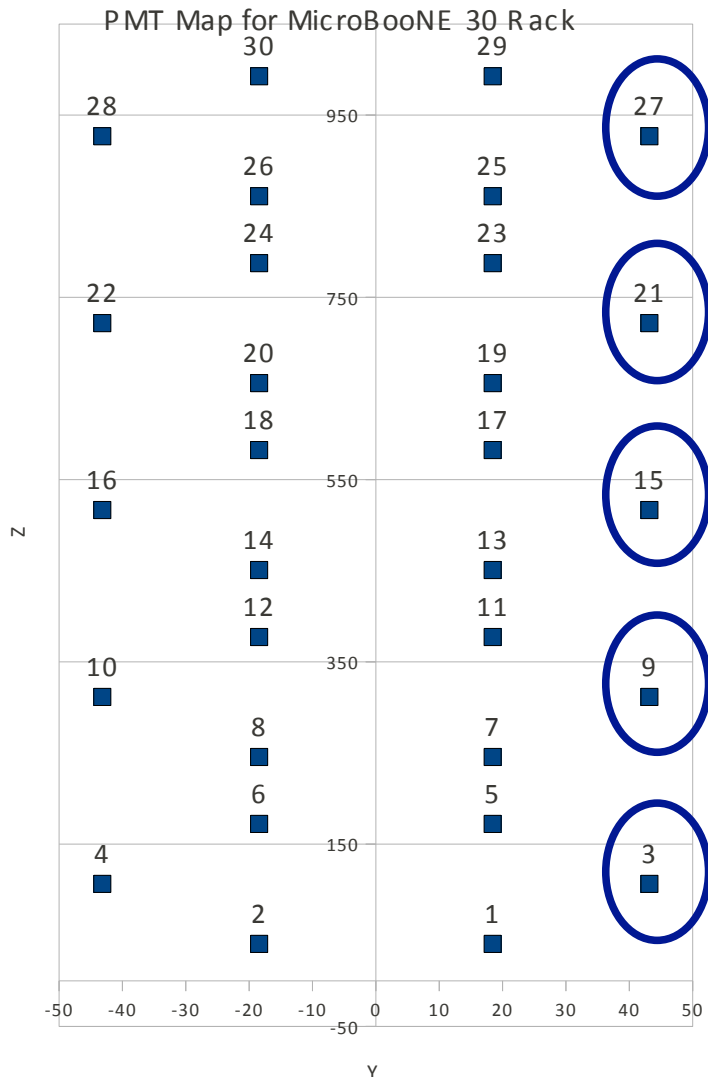


**Ask two questions:**

**1) How wide is the coverage of each PMT?**

**2) How does the global coverage change if a given PMT fails?**

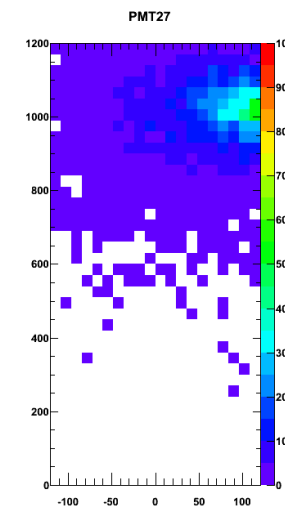
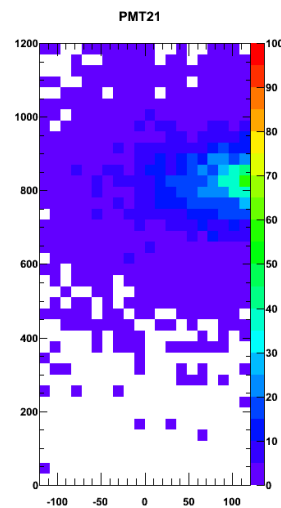
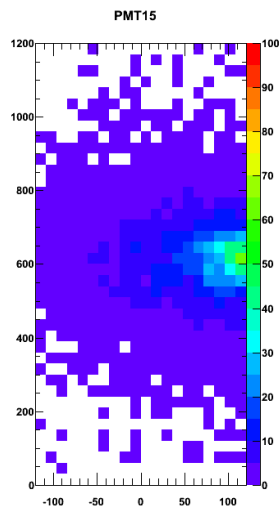
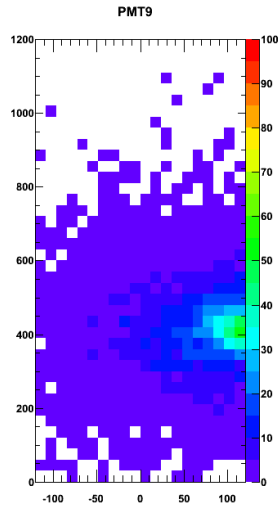
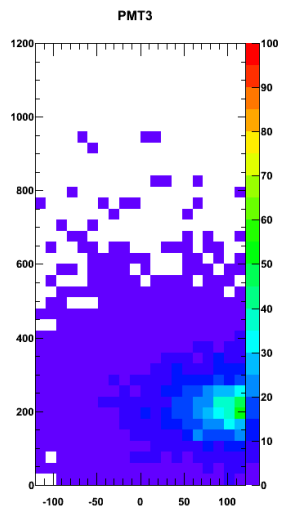
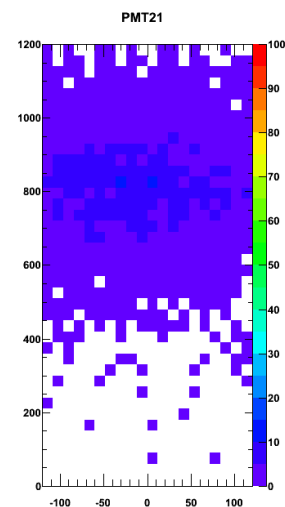
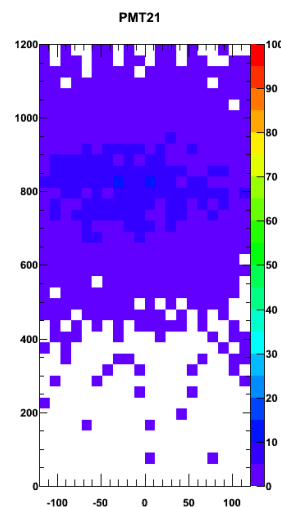
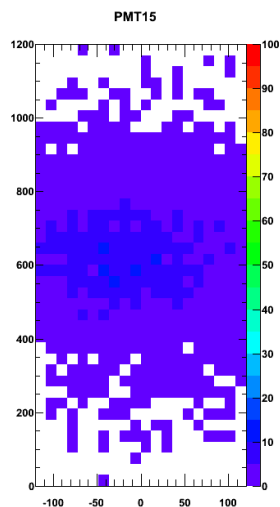
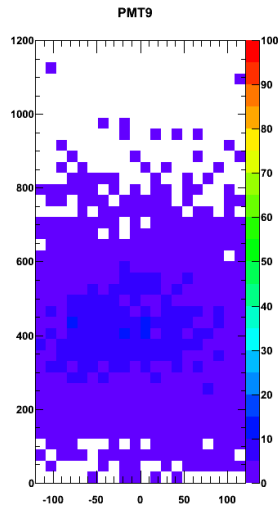
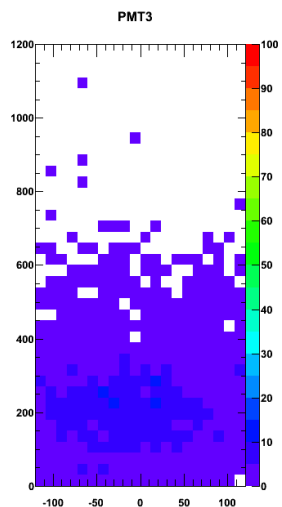
# PMT Coverage and Redundancy Tests



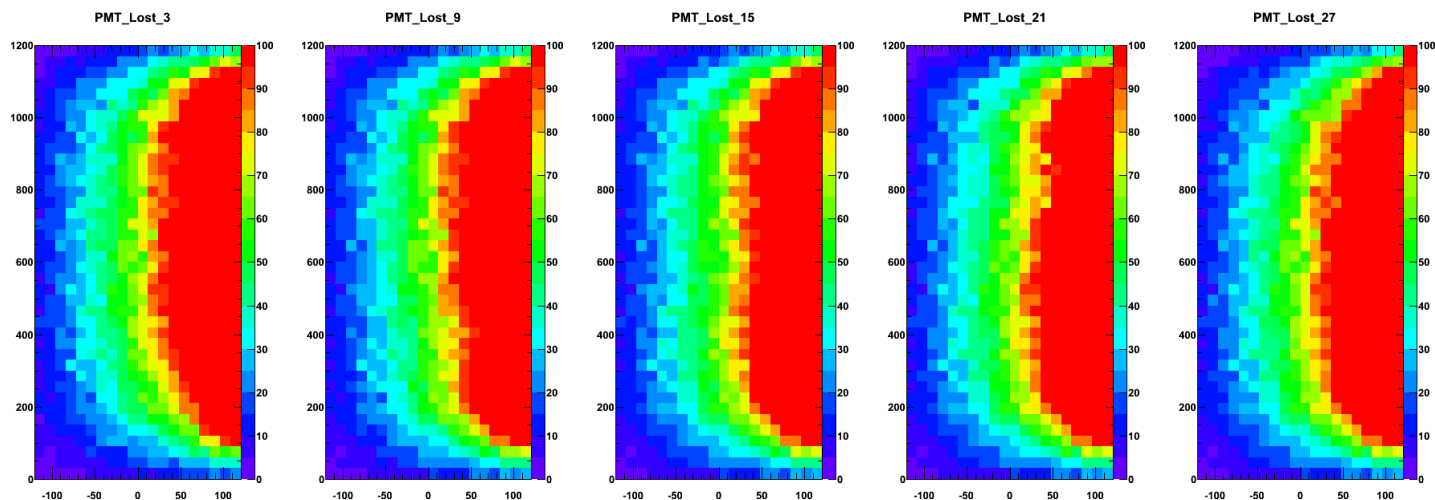
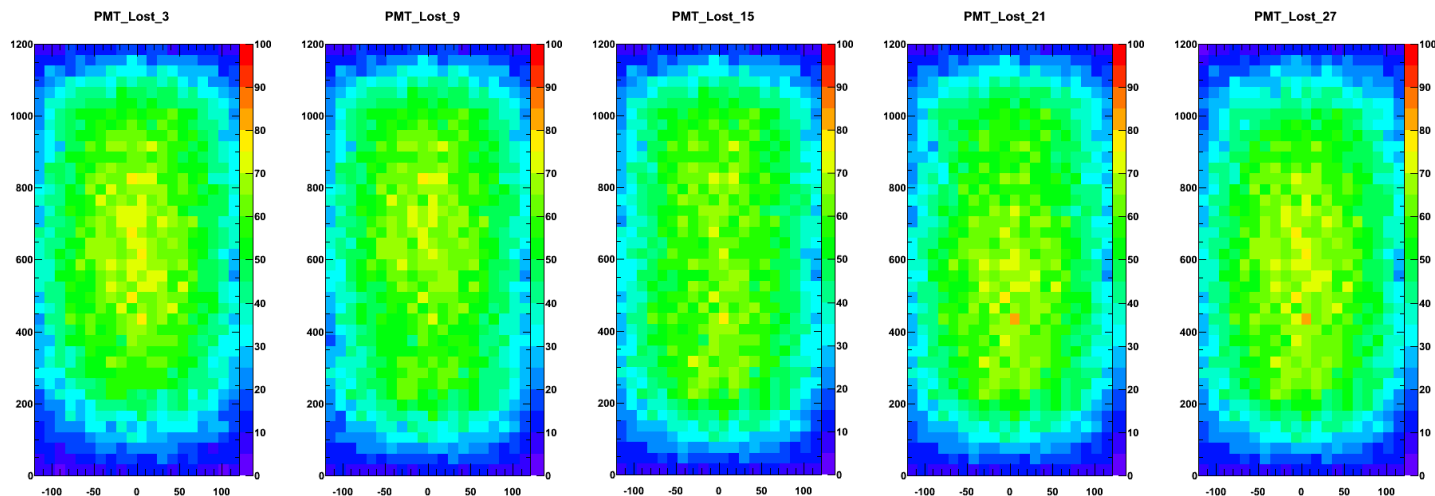
**As an example:**

**consider a line of PMTs in Z**

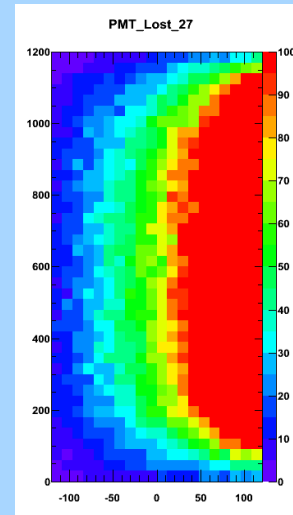
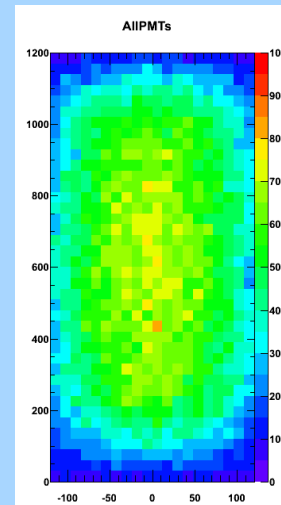
# PMT Coverage Test



# PMT Redundancy Test



ONE PMT MISSING



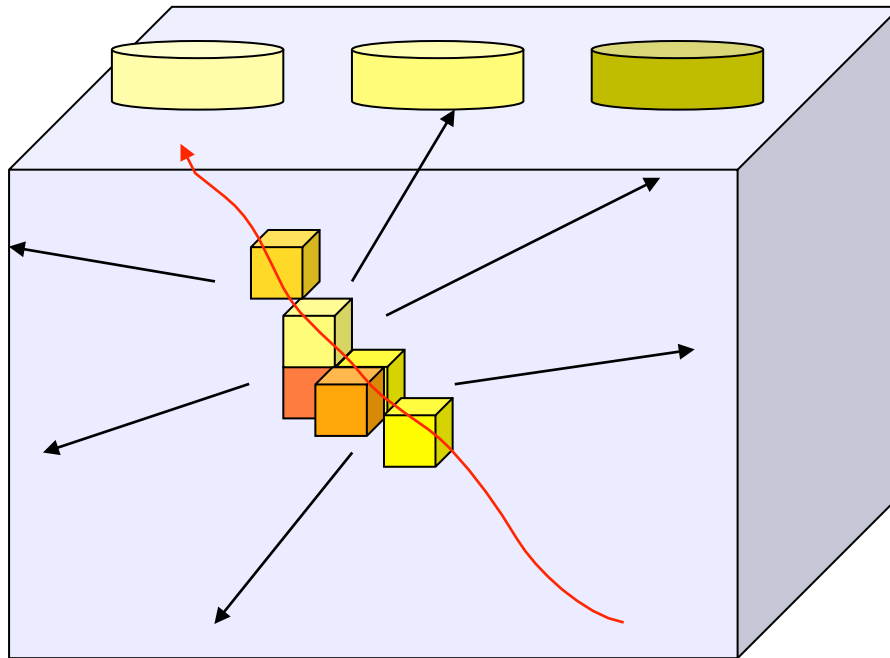
ALL PMTS



# 5 – Fast Simulation

# Fast Simulations and Photon Library Sampling

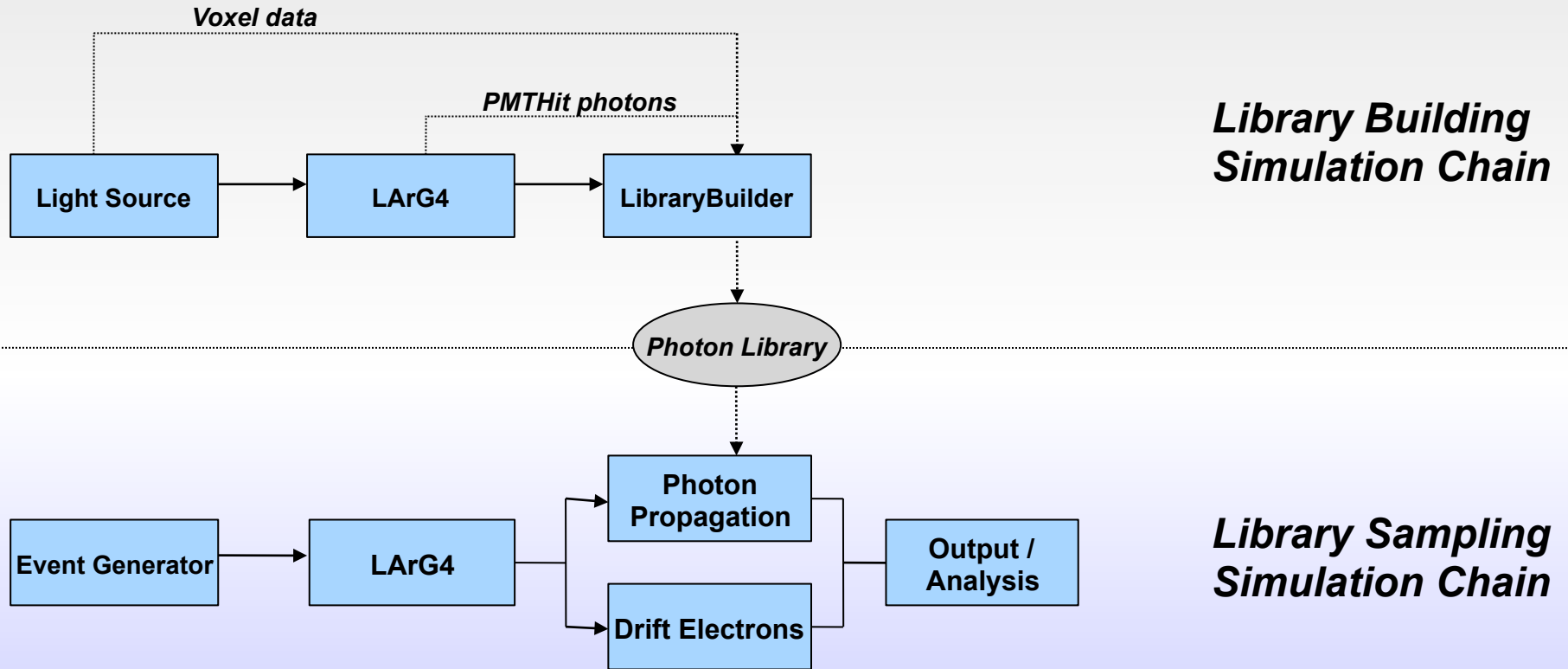
- GEANT4 simulation of 100,000s of photons per event takes a very long time – not a feasible approach for long monte carlo runs
- Scintillation photons are produced isotropically and in large numbers so we can take a different approach and sample from a library of typical responses



*How many photons from each "voxel" will reach each PMT?*

*How will their angles of incidence and positions on the PMT face be distributed?*


# Voxelized PhotonLibrary Building And Sampling



















- Library is built using a light source with gaussian spectrum of **9.7 +/- 1 eV** in each voxel
- Later sampled by new module **PhotonPropagation**, which runs in parallel with DriftElectrons
- During the LarG4 step of the sampling chain, we do not step any photons, simply provide the **number produced in each voxel**

# PhotonPropagation Code

root / trunk / PhotonPropagation

 Statistics | Revision:

Name	Size	Revision	Age	Author	Comment
 GNUmakefile	1 KB	<a href="#">848</a>	about 1 year	Brian Rebel	make build order a thing of the past
 LinkDef.h	518 Bytes	<a href="#">487</a>	over 1 year	Benjamin Jones	First commit of fastsim photon library builder
 PMTResponseAnalyzer.cxx	8.7 KB	<a href="#">2043</a>	7 months	Brian Rebel	missed one commit on the SimListUtils conversio...
 PMTResponseAnalyzer.h	3 KB	<a href="#">1520</a>	11 months	Brian Rebel	convert to using RandomNumberGenerator rather t...
 PMTResponseAnalyzer_module.cc	576 Bytes	<a href="#">587</a>	over 1 year	Brian Rebel	changes for the full ART rollout
 PhotonLibrary.cxx	2.3 KB	<a href="#">1310</a>	about 1 year	Brian Rebel	quiet compiler warning
 PhotonLibrary.h	1.4 KB	<a href="#">587</a>	over 1 year	Brian Rebel	changes for the full ART rollout
 PhotonLibraryBuilder.cxx	2.9 KB	<a href="#">2043</a>	7 months	Brian Rebel	missed one commit on the SimListUtils conversio...
 PhotonLibraryBuilder.h	1.1 KB	<a href="#">1520</a>	11 months	Brian Rebel	convert to using RandomNumberGenerator rather t...
 PhotonLibraryBuilder_module.cc	592 Bytes	<a href="#">587</a>	over 1 year	Brian Rebel	changes for the full ART rollout
 PhotonLibraryObjects.cxx	824 Bytes	<a href="#">1105</a>	about 1 year	Brian Rebel	more doxygen fixes
 PhotonLibraryObjects.h	1.1 KB	<a href="#">587</a>	over 1 year	Brian Rebel	changes for the full ART rollout
 PhotonLibraryService.cxx	5 KB	<a href="#">587</a>	over 1 year	Brian Rebel	changes for the full ART rollout
 PhotonLibraryService.h	1.3 KB	<a href="#">587</a>	over 1 year	Brian Rebel	changes for the full ART rollout
 PhotonLibraryService_service.cc	522 Bytes	<a href="#">587</a>	over 1 year	Brian Rebel	changes for the full ART rollout
 photpropmodules.fcl	580 Bytes	<a href="#">1973</a>	7 months	Christopher Green	Updates for ART v1.0

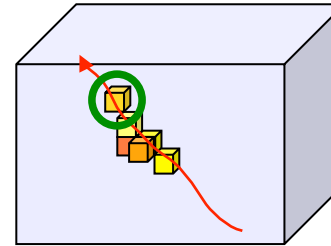
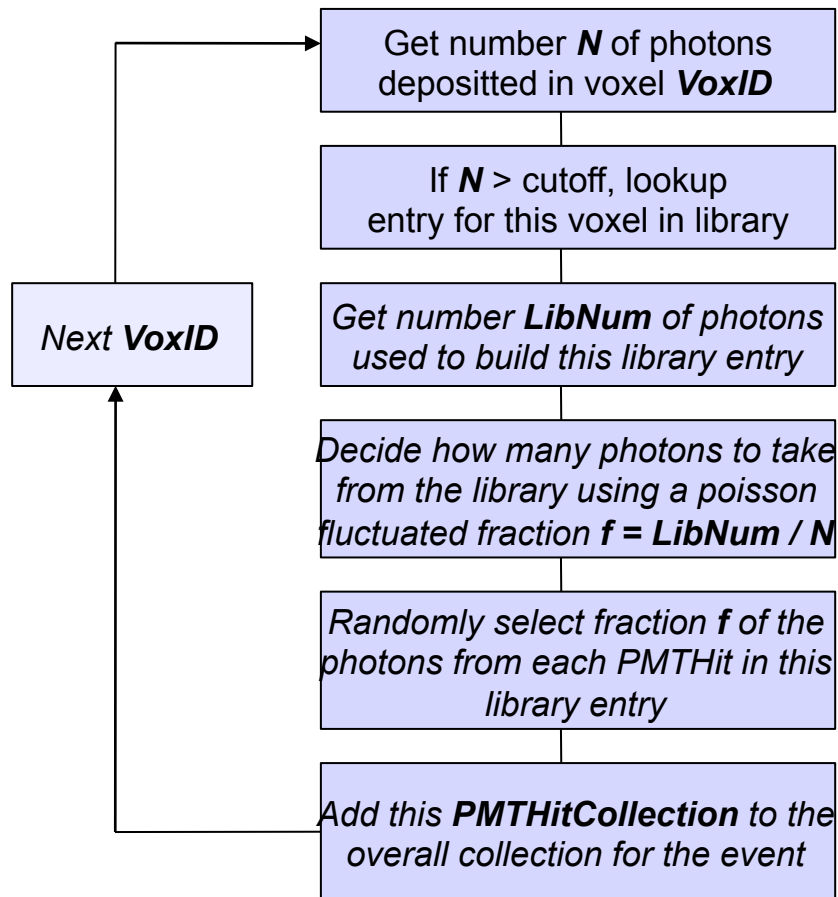
Latest revisions

# Mechanics of Library Sampling: LArG4

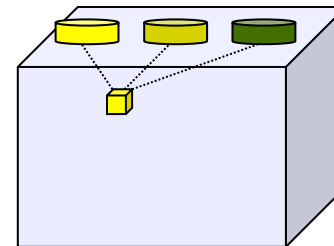
***Note: This code not yet in the LArSoft repository.***

- When running the Geant4 part of the simulation, a new physics constructor, FastOpticalPhysics is used in place of the standard OpticalPhysics
- This is identical, with the exception that the process G4Scintillation is replaced with the FastScintillation process. This process increments a counter for each optical voxel when a photon is produced inside, rather than adding an OpticalPhoton to the stack
- The voxelized light information is stored in a sim::PhotonProduction object in the event.
- Cerenkov photons are still stepped in the usual way.

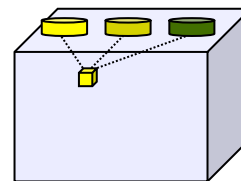
# Voxelized Library Sampling



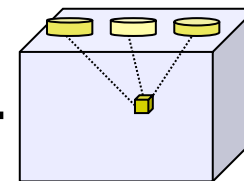
**LArG4**



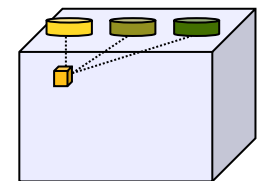
**PhotonLibrary**



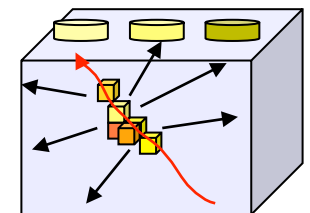
+



+



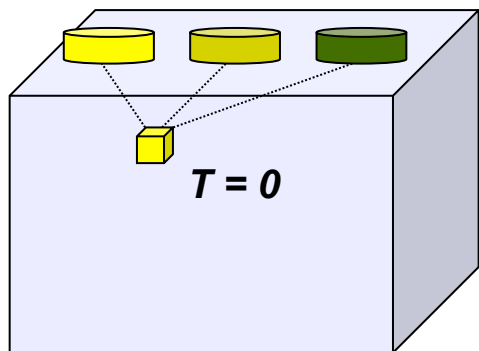
=



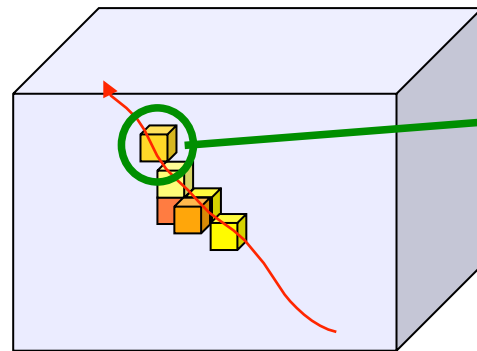
**Photon Propagation Module**

# Time Smearing for Fast Sim

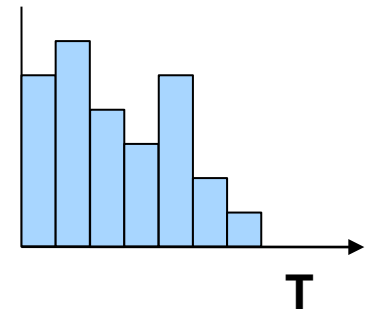
- One of the main tasks of the PMT's is triggering – here we require a coincidence with the beam window. Hence timing is important for trigger studies.
- Library entries are generated by photons at time  $t = 0$ .
- Accompanying **PMTHits** store a **time value** for each photon which reached the PMT
- As well as reading # of photons per voxel, store a time profile for that voxel. Then time smear PMT hits sampled from library using this profile.



Library

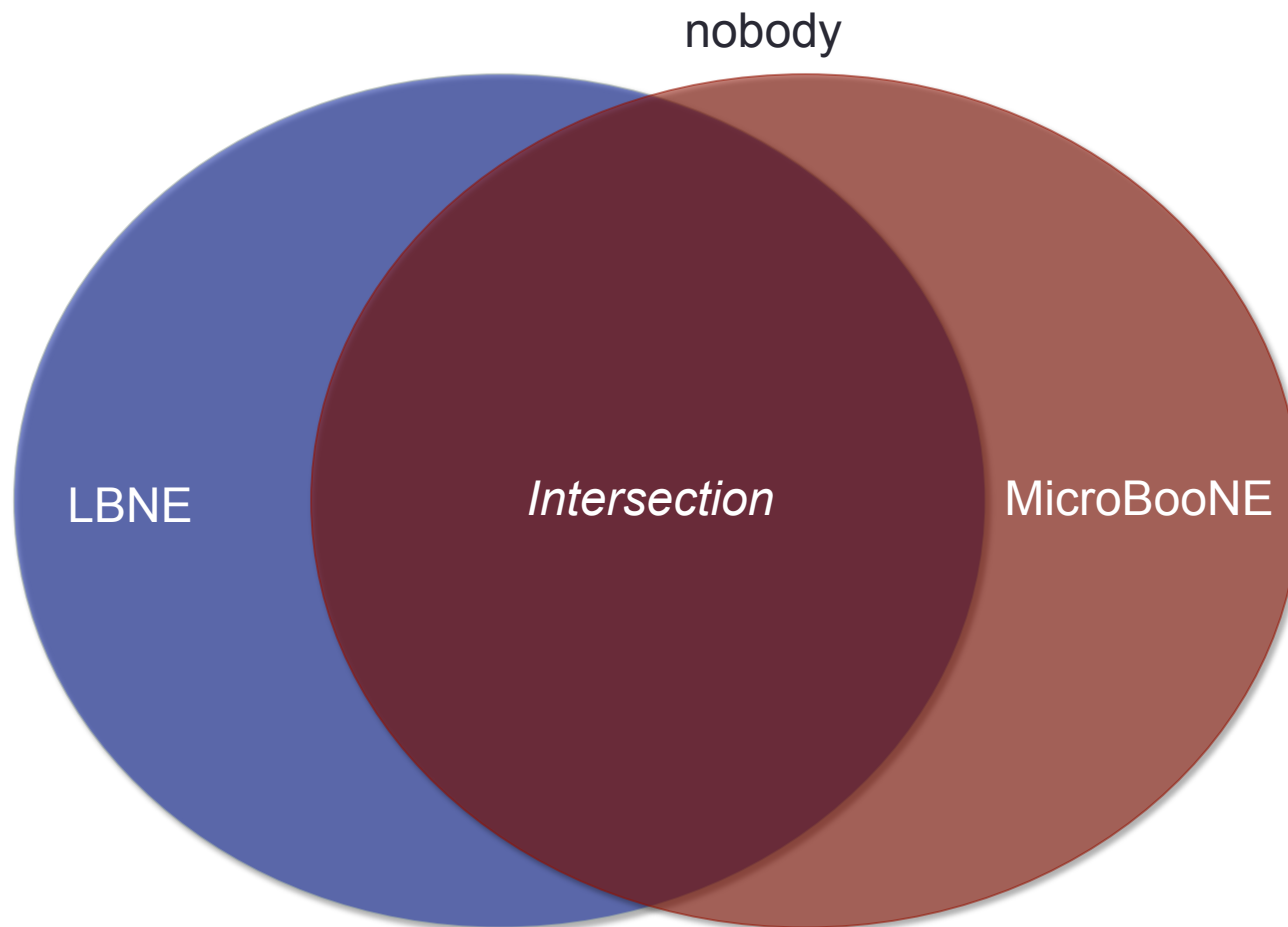


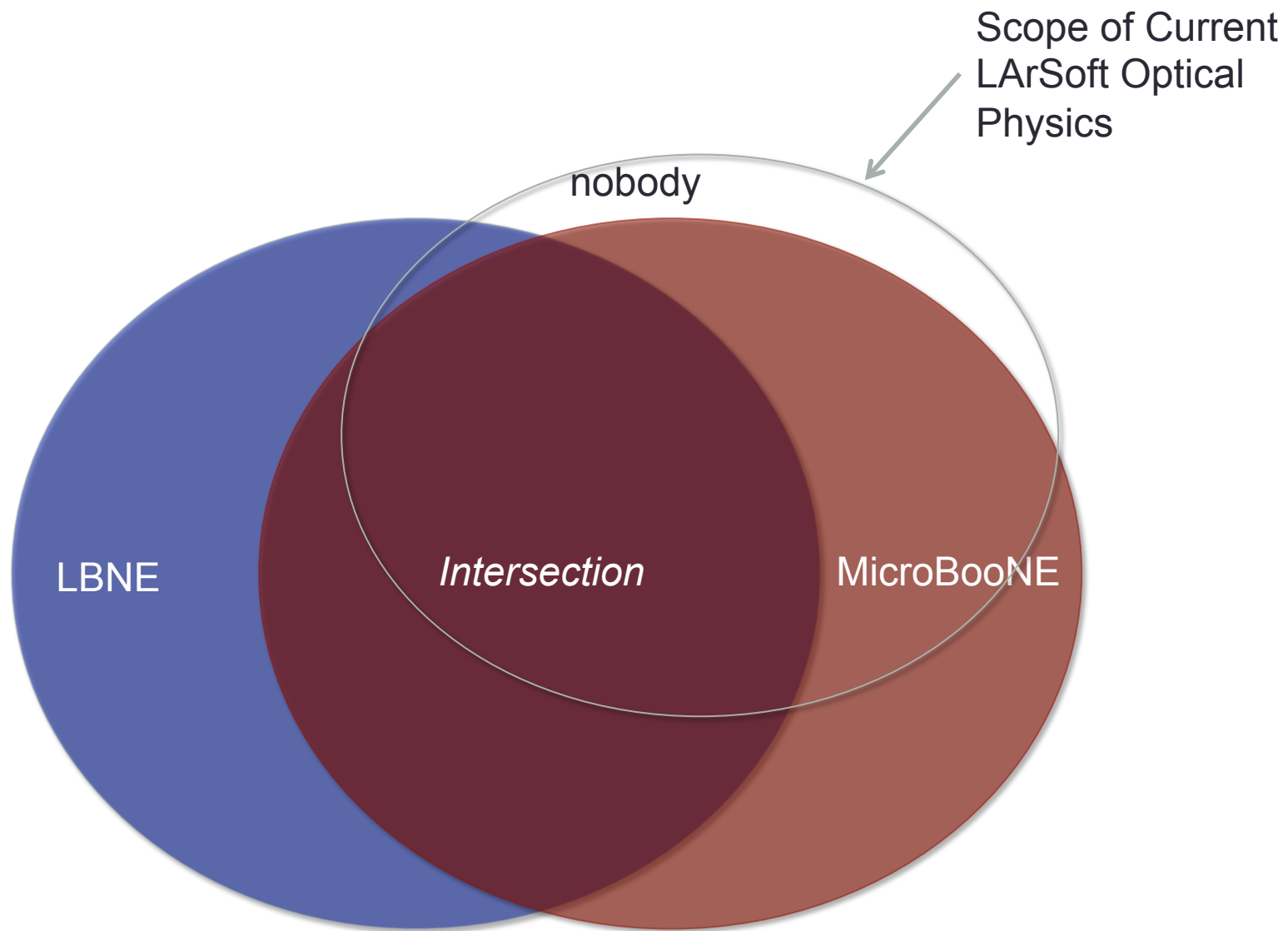
LArG4



# 6 – What MicroBooNE Needs to Do, and What LBNE Needs To Do

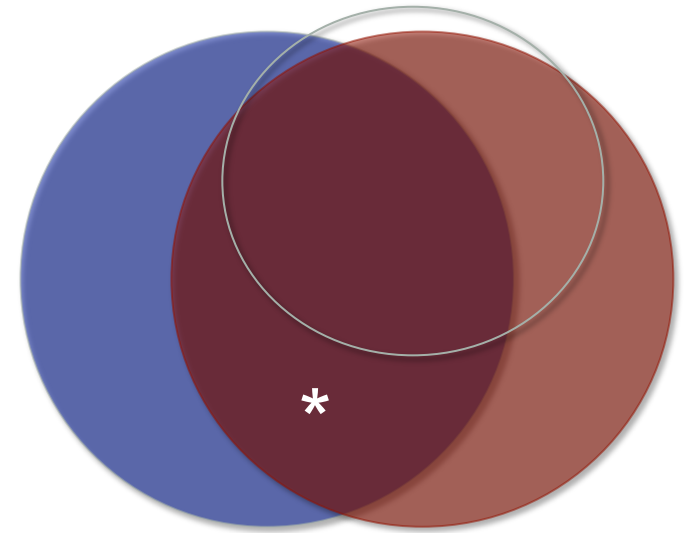




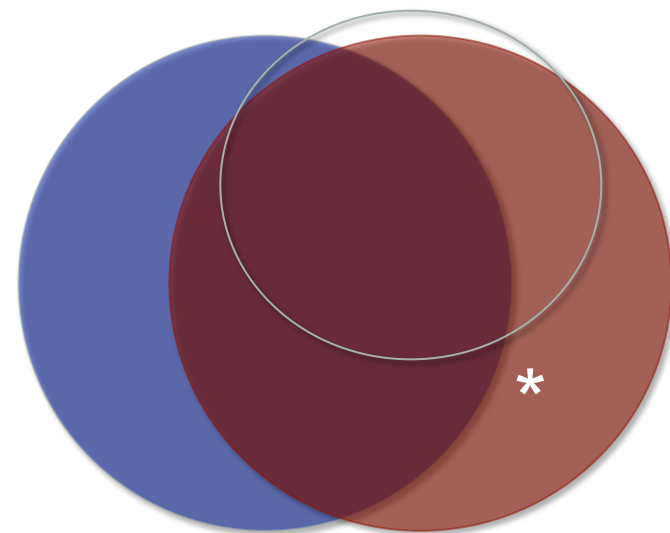


# Both Need

- Completed fast optical sim
  - My priority from Sept-Dec 2012
- Cosmic discrimination with scintillation
  - Which I expect to get involved in at some point
- Some connection to trigger objects
  - But they currently don't exist in LArSoft...
- Realistic quenching, parameterization of purity and temperature dependent effects
  - A hard problem, possibly solvable by NEST

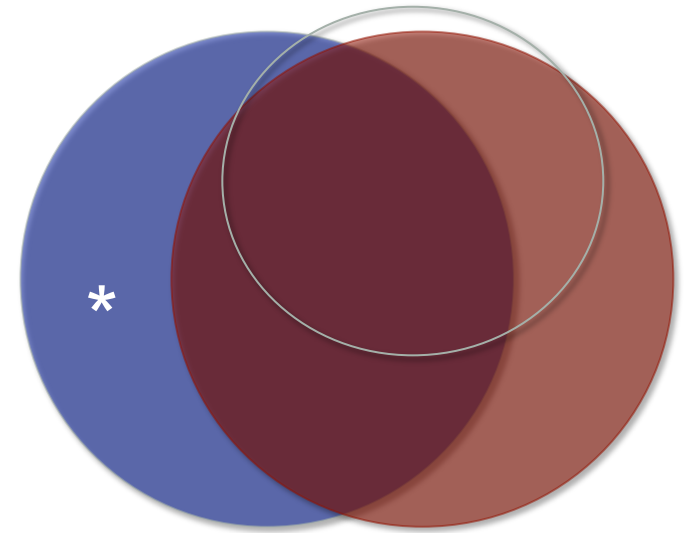


# MicroBooNE Needs



- PMT assemblies with realistic quantum efficiency / angular dependence, etc
- Digitization routines, accounting for PMT pulses, noise and nonlinearities
- Better parameters for surface reflectances
  - All these require data – we expect to learn a lot with test stands this summer

# LBNE Needs



- Optical system placement geometry
- Local geometry of each sensitive paddle object
- Digitization routines for paddles
- Maybe an even faster fast sim
  - (library building may be unfeasible for such a big detector.)
- Additional levels of data structure for many paddles to one PMT, or many PMTs to one TPC, etc

# 7 – Hooks for Alternative Treatments

# NEST, etc

- There have been several approaches to improving the parameterization of scintillation in LAr, the latest being NEST
- I believe that interfacing these approaches with LArSoft should be reasonably straightforward – the key is the configurable physics list

# Step 1: New Physics Constructor

- Follow the example of `OpticalPhysics.cxx` and define a new physics constructor, say, ***NESTOptical***, to interface with the `ConfigurablePhysicsList`
- In this object, there will be the declaration of a process to create optical photons, analogous to ***G4Scintillation***. This process will contain all of the NEST physics
- Then, `NESTOptical` can be swapped in or out of the physics list at runtime



## Step 2 – Fast Sim Interface

- Then, if you also want to use this physics to feed the fast propagation module, need another physics object which produces PhotonProduction objects instead of adding photons to the stack
- If this becomes more than a one-time use case then maybe we need a cleverer way of treating photons which can figure out which simulation path to put them through
- Discussion? (or maybe best left for a LArSoft meeting)

# Summary

- A detailed optical simulation for scintillation and Cerenkov light in liquid argon has been developed within the LArSoft framework
- Optical system geometry definition scripts and a specific PMT geometry for the MicroBooNE detector have been implemented
- Preliminary studies using simulated light sources suggest ample PMT coverage for triggering at all detector locations except the corners of the TPC with significant redundancy
- A parameterized fast optical sim is being implemented and will eventually form part of the standard MicroBooNE simulation chain
- LBNE and MicroBooNE have work to do, I intend on doing some of it – simulation based aspects. Need "volunteers" for the rest – digitization and trigger related aspects